**SMA与EMA：talib与中国股票行情软件的差异**

### **一、引言**

在股票行情软件的公式中，SMA、EMA经常出现，比如:  
KDJ(N,M,P)在我用的“交易师”软件中定义如下：

RSV:=(CLOSE-LLV(LOW,N))/(HHV(HIGH,N)-LLV(LOW,N))\*100;

K:SMA(RSV,M,1);

D:SMA(K,P,1);

J:3\*K-2\*D;

再看MACD(S,P,M)的定义：

DIFF : EMA(Close,S) - EMA(Close,P);

DEA : EMA(DIFF,M);

MACD : 2\*(DIFF-DEA);

SMA是中国式的移动平均，英文名姑且叫Smooth Moving Average吧，  
EMA的英文名Exponential Moving Average。

python中做各种平均计算时，我们经常引用ta-lib库。ta-lib支持的平均种类如下：

MA\_Type: 0=SMA, 1=EMA, 2=WMA, 3=DEMA, 4=TEMA, 5=TRIMA, 6=KAMA, 7=MAMA, 8=T3 (Default=SMA)

许多新手，就误以为ta-lib里面的SMA跟中国SMA是等同的。

其实ta-lib里面的SMA是Simple Moving Average，等同于Excel里面的Average，等同于python的mean()，等同于中国股票行情软件的MA.

### **二、算法**

中国SMA的算法：

若Y=SMA(X,N,M) 则

Y=(M\*X+(N-M)\*Y')/N, 其中Y'表示上一周期Y值,N必须大于M

EMA的算法：

若Y=EMA(X,N) 则

Y=[2\*X+(N-1)\*Y']/(N+1), 其中Y'表示上一周期Y值。

由以上两个公式可以看出：

* SMA、EMA计算本期的均值，都引用了前一期的均值。那前一期再引用前一期的前一期，如此向前引用，岂不父辈、祖辈....无穷尽也？
* SMA可以替代EMA，即

EMA(X,N)=SMA(X,N+1,2)

### **三、实践与验证**

以603536.XSHG惠发股份为例, 根据SMA的定义，计算SMA(C,18,1)

Y = (1\*X+(18-1)\*Y')/18 = (X+17\*Y')/18

yn=xn+17yn−118yn=xn+17yn−118

#### **用Excel验证**

| **A** | **B** | **C** | **D** | **E** |
| --- | --- | --- | --- | --- |
| 1 | 日期 | 收盘价C | SMA(C,18,1) | D列的Excel公式 |
| 2 | 2017/6/13 | 10.99 | 10.990 | =C2 |
| 3 | 2017/6/14 | 12.09 | 11.051 | =(C3+17\*D2)/18 |
| 4 | 2017/6/15 | 13.3 | 11.176 | =(C4+17\*D3)/18 |
| 5 | 2017/6/16 | 14.63 | 11.368 | ....... |
| 26 | 2017/7/17 | 20.27 | 19.183 |  |
| 27 | 2017/7/18 | 19.64 | 19.208 | ...... |

#### **验证结果**

* 实际计算的时候，是从前往后顺推的，所以不存在递归无穷尽的问题。
* 计算结果与“交易师”行情软件计算结果一致。

#### **测试**

****在“交易师”股票行情软件****，验证一下：  
**EMA(C,18)**与**SMA(C,19,2)**的结果是一致的。****

### **四、python的实现**

#### **SMA**

**def** sma\_cn1(X, n, m):

**return** reduce(**lambda** a, b: ((n - m) \* a + m \* b) / n, X)

如果你觉得reduce晦涩难懂，可以自己循环：

**def** sma\_cn2(X, n, m):

y = X[0]

**for** i **in** range(1, len(X)):

y = (m \* X[i] + (n - m) \* y) / n**展开代码 ↓**

可以自己验证，以上两种写法，计算结果完全相同。但需要注意两个问题：

* X里面如果有nan值的时候，需要先行处理
* 以上两种写法，是array输入，单值输出。按照ta-lib的习惯，****array in, array out****,那你还需要改造一下这个函数。

#### **EMA**

* 引用SMA

**def** my\_EMA(X,n):

**return** sma\_cn(X,n+1,2)

* 引用talib

**import** **talib**

**def** talib\_EMA1(X, n):

**return** talib.EMA(X, n)**展开代码 ↓**

### **五、万事大吉？**

#### **No**

你以为你弄懂了这些算法，python的实现，就万事大吉了，****NO！****

#### **坑你一下**

X = np.array(range(30))\*1.0**print** 'talib\_EMA1:'**print** talib\_EMA1(X, 18)**print** 'talib\_EMA2:'**print** talib\_EMA2(X, 18)**展开代码 ↓**

talib\_EMA1:

[ nan nan nan nan nan nan nan nan nan nan nan nan

nan nan nan nan nan 8.5 9.5 10.5 11.5 12.5 13.5 14.5

15.5 16.5 17.5 18.5 19.5 20.5]

talib\_EMA2:展开输出 ↓

#### **居然还是不一致**

****EMA(X, n)居然跟sma\_cn(X, n+1, 2)不一样啊！****  
你不是验证过，EMA(X,n)与sma\_cn(X,n+1,2)是一致的么？呵呵，你多心了，我验证的时候，加上了一句话：  
****在中国的股票行情软件里二者是一致的****  
并没有说talib\_EMA与中国EMA是一致的！  
****如果你想当然地以为ta-lib的EMA与中国的EMA是一致的，那又错了。****

#### **差异的原因**

看看以上输出，talib\_EMA系列在输出的时候，前面有17个nan，第一个非nan的值是8.5，那这个8.5是从哪里来的？

(0+1+2+....+17)/18.0 = 153/18.0 = 8.5

是前18个数的算术平均值。

#### **talib-EMA(X,n)的算法是：**

**def** copy\_talib\_EMA(X, n):

Y = np.array([np.nan] \* len(X)) *# 初始化为全nan数组*

Y[n - 1] = X[:n].mean()

**for** i **in** range(n, len(X)):

Y[i] = (2 \* X[i] + (n - 1) \* Y[i - 1]) / (n + 1)**展开代码 ↓**

**print** 'talib\_EMA1:'**print** talib\_EMA1(X, 18)**print** 'copy\_talib\_EMA:'**print** copy\_talib\_EMA(X, 18)

talib\_EMA1:

[ nan nan nan nan nan nan nan nan nan nan nan nan

nan nan nan nan nan 8.5 9.5 10.5 11.5 12.5 13.5 14.5

15.5 16.5 17.5 18.5 19.5 20.5]

copy\_talib\_EMA:展开输出 ↓

### **结论**

如果你很在意计算结果跟中国股票行情软件的一致性，那SMA请你使用sma\_cn函数，EMA一律用sma\_cn(X,n+1,2)替代。  
不要用ta-lib的EMA，当然更不能用ta-lib的那个“冒牌”SMA。